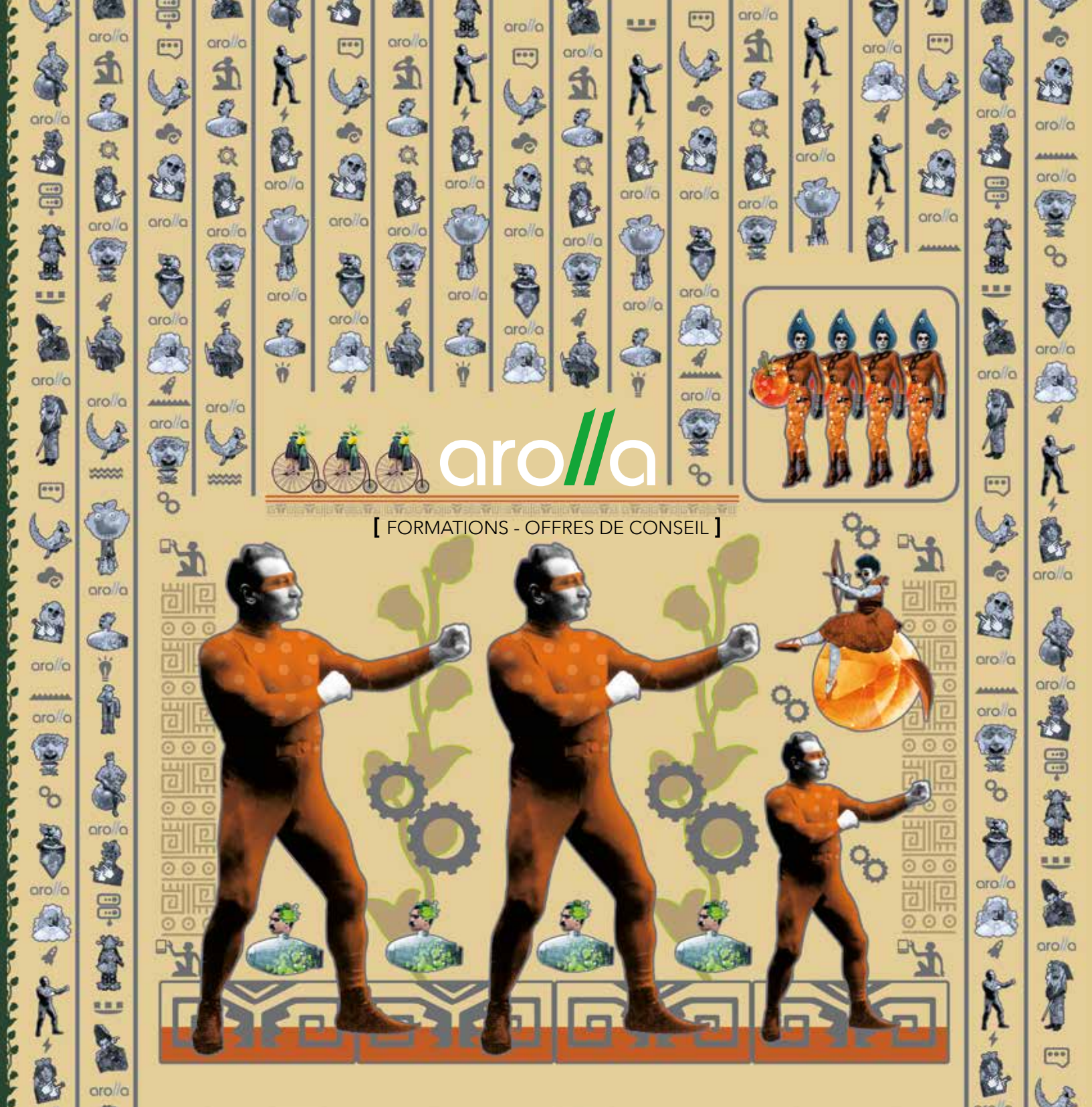

arolla
[FORMATIONS - OFFRES DE CONSEIL]

[INSTITUT DE FORMATION - OFFRES DE CONSEIL]

Edito

Depuis sa création en 2009, Arolla s'attache à promouvoir les pratiques de développement issues du Software Craftsmanship. Nos formations reflètent notre ligne éditoriale et abordent sous divers formats, de 1 à 3 jours, l'état de l'art : **le TDD et le Clean Code (SOLID), Behavior-Driven Development (BDD), Domain-Driven Design (DDD) et l'Optimisation de code legacy**. Nous avons complété notre offre avec un ensemble de formations de design et d'architecture. Notre savoir-faire accumulé chez nos clients nous a permis de monter l'offre conseil « Azure Continuous Delivery » en collaboration avec Microsoft, ce qui nous permet, donc, de vous proposer une formation Azure.

Que ce soit dans le cadre d'interventions au sein des équipes de développement d'entreprises de tous les secteurs, de missions de conseil ou de formations, les Software Gardeners d'Arolla s'attachent à diffuser consciencieusement les pratiques les mieux adaptées à chaque contexte pour améliorer la qualité du code et donc être plus efficace dans la durée. Le développement logiciel est un sport d'endurance !

Nous militons pour diffuser les pratiques les plus utiles du Software Craftsmanship au-delà des cercles de développeurs avant-gardistes, au bénéfice de notre industrie.

Sommaire

NOS FORMATIONS

		DURÉE	PAGE
100% CRAFT PAR AROLLA			
> Craftsmanship & tests unitaires		2 jours	p. 12
> Code retreat	Date unique en intra-entreprise	Contactez-nous	p. 14
> TDD – Test Driven Development		2 jours	p. 16
> Working On Legacy Code		2 jours	p. 18
> BDD – Behaviour Driven Development		2 jours	p. 20
> DDD – Domain Driven Design par Cyrille MARTRAIRE		2 jours	p. 22
> DDD – Domain Driven Design		2 jours	p. 24
> Crafting Front-End Code		2 jours	p. 26
PASSEZ À LA PROGRAMMATION FONCTIONNELLE			
> Programmation fonctionnelle en F#		2 jours/3jours	p. 28
> Programmation fonctionnelle en JAVA		2 jours	p. 30
BOOSTEZ VOTRE DESIGN, VOTRE ARCHITECTURE ET VOTRE INFRA			
> Microservices avec DDD		2 jours	p. 32
> Living Documentation par l'auteur		1 jour	p. 34
> Pragmatic Architecture		2 jours	p. 36
> Premiers pas dans Azure		3 jours	p. 38

NOS OFFRES DE CONSEIL

	PAGE
> Coaching Embarqué	p. 42
> Un audit constructif	p. 43
> Stratégie de test	p. 44
> Tester l'intestable	p. 45
> Faire converger les perspectives	p. 46
> Aller vers des microservices à partir d'un monolithe legacy	p. 47
> Définir les tracés directeurs	p. 48
> Faire grandir les équipes, renforcer les champions internes	p. 49
> Booster une initiative, valoriser les savoirs & les expériences	p. 50
> Rendre lisible la complexité	p. 51
> Création d'outils spécifiques	p. 52
> Accompagner votre croissance au-delà des changements d'échelle	p. 53
> Accompagner une transformation d'envergure	p. 54

Toutes nos formations sont d'une durée variant de 1 à 3 jours. Elles proposent un contenu entièrement personnalisable. Elles peuvent être réalisées, à la demande, dans nos locaux dédiés au 21, rue du Bouloi Paris 1er - Les Halles.

Elles comportent toute une partie théorique complétée autant que possible par des échanges interactifs, des exercices et des katas.

Nous préconisons de limiter chaque session à 10 participants environ.

Sur demande, les formations peuvent être données partout en France et à l'étranger.

Inscription :

- ↘ **en ligne sur www.arolla.fr/training**
- ↘ **par e-mail : contact@arolla.fr**
- ↘ **par téléphone : 01 77 38 82 00**
- ↘ **par courrier : Arolla, 25 rue du Louvre - 75001 PARIS**

[Arolla Training]

Pourquoi suivre une formation Arolla ?

Au sein d'Arolla, nous sommes convaincus de l'importance de la formation et de l'apprentissage continu. Chaque année, nos consultants disposent d'un budget qui leur permet de financer et de parfaire leur savoir-faire à la source : lors de conférences en France ou à l'étranger, en participant à divers coding dojos ou enfin, en suivant des trainings sur des sujets relatifs à notre ligne éditoriale.

Nous sommes persuadés que c'est bien l'entraînement et la pratique régulière qui font progresser les développeurs. Pour ces raisons, nous valorisons particulièrement les formations. Celles-ci sont constituées au minimum de 50% d'exercices concrets, qu'on prénomme aussi katas.

En tant qu'institut de formation, Arolla se focalise sur des sessions qui vous permettront d'améliorer vos techniques de développement. Ces formations sont proposées aussi bien en inter entreprises (dans nos locaux, avec des stagiaires issus de différentes sociétés) qu'en intra entreprises (l'ensemble des stagiaires est salarié de la même société et la formation se déroule en son sein ou dans les locaux d'Arolla).

Nous proposons également des formations sur mesure. N'hésitez donc pas à nous solliciter pour que nous mettions en place la solution la plus adaptée à vos besoins.
(contact@arolla.fr ou 01 77 38 82 00)

[Arolla Training]

Formations intra/ sur mesure/Conseil

Chez Arolla, nous aimons aussi travailler sur mesure ! Après avoir échangé avec vous sur les contours de votre besoin, nous saurons vous proposer rapidement :

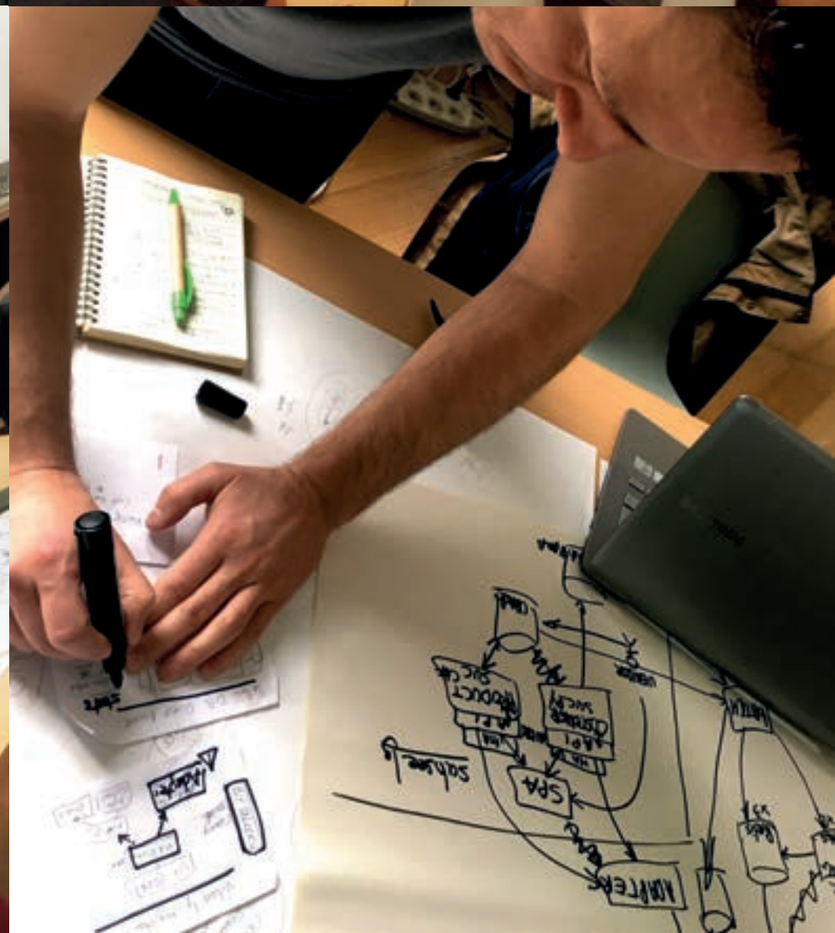
- Des plans de cours personnalisés, sur les bases d'un existant ou complètement rédigés pour vous.
- Des trainings dispensés dans vos locaux (à Paris, en France ou à l'étranger) dans nos locaux dédiés à la formation, dans le premier arrondissement de Paris.
- En fonction de vos souhaits, l'assistance ne sera ouverte qu'à vos collaborateurs ou complétée avec des stagiaires d'horizons différents pour optimiser vos coûts.
- Des trainings animés en anglais ou en français, dans l'environnement technique de votre choix (nous sommes experts sur Java et C#, pour tout autre langage nous consulter).
- Des tarifs forfaitaires tenant compte du volume d'inscrits.

Nos trainings peuvent être complétés par une ou plusieurs missions de coaching technique dans le cadre desquelles un coach viendra accompagner vos développeurs sur des problématiques « de la vraie vie » et sur du code de production (cf. catalogue de conseil). Cela donnera l'occasion de mettre en pratique le savoir-faire qui aura été acquis lors de nos trainings.

Nous vous fournirons les supports de cours en version dématérialisée à l'issue de la formation.

Dans l'hypothèse où le training aurait lieu dans vos locaux, nous vous demanderons simplement de nous mettre à disposition le matériel nécessaire au bon déroulé de la formation (ordinateurs, vidéo-projecteur, IDE, runners de tests...).

Nos locaux dédiés à la formation du 21 rue du Bouloi (Paris 1er – Les Halles, Louvre Palais-Royal) sont équipés pour pouvoir accueillir une quinzaine de stagiaires dans de très bonnes conditions.



[Formations]

[Formations]

Craftsmanship et Tests Unitaires

2 jours	**Méthodes pédagogiques :** 50% théorique, 50% pratique (exercices à base de code)
Prochaines sessions : retrouvez les dates et le calendrier complet sur notre site www.arolla.fr/training	
Offre inter-entreprise : Petit-déjeuner - Déjeuner - Café et collation	

Objectifs :

Au cours de cette formation vous allez :

- **Etre sensibilisé à l'écriture des tests**
- **Apprendre à écrire des vrais tests unitaires (TU)**
- **Ecrire des tests lisibles et maintenables**

Aujourd'hui admis par la majorité des développeurs, (presque) tout le monde écrit des tests dits unitaires. Mais il nous reste à savoir comment écrire de bons tests unitaires, car – comme chez les chasseurs – on trouve des bons et des mauvais TUs.

Qu'est-ce qu'un test unitaire ? Comment l'écrire ? Faut-il ou non utiliser des mocks ? Doit-on tester une méthode privée ? Cette formation est là pour que vos tests unitaires ne deviennent pas des boulets aux pieds de votre projet et pour que vous puissiez atteindre votre ROI sur cet investissement.

Programme :

Tests unitaires

- Définition, présentation théorique
- Pourquoi écrire des tests
- Vérifier les tests, présence d'assertions
- Détecter et éviter les régressions
- Mettre en évidence la raison d'un test en échec, par le nommage des tests et des assertions explicites
- Documenter son code par les tests, l'importance du vocabulaire et de la granularité des tests par règle métier
- Refactorer les tests, et les critères pour supprimer certains tests
- Mise en pratique avec un Kata en pair-programing en ping-pong

Mocks

- Présentation théorique : c'est quoi un mock ? (fakes: mocks & stub)
- Quand utiliser un mock : collaborateurs vs dépendances
- Ne pas figer les détails d'implémentations (utilisations des any, etc.)
- Principe du "tell don't ask"
- Mise en pratique des approches de mock avec un kata classique adapté

Public :

Vous êtes développeur junior ou intermédiaire à l'aise avec votre langage de programmation.

Pré-requis :

- Connaître un langage de programmation
- Les participants doivent avoir un poste de travail avec un environnement de développement avec un framework de test.

Matériel pédagogique :

Le formateur distribuera les supports de la formation au format électronique PDF à la fin de la formation.

Code Retreat

1 jour	**Méthodes pédagogiques :** 100% pratique
Date unique en intra-entreprise	
Cette formation peut avoir lieu en intra-entreprise. Nous contacter pour plus d'informations	

Objectifs :

Offrir un espace de réflexion pour :

- **Expérimenter des façons différentes de coder, tester, faire du design et collaborer en échangeant avec des pairs**
- **Découvrir ou approfondir la pratique du Test-Driven Development (TDD) et du Pair-Programming**
- **Permettre de lever la tête des projets**
- **Renforcer la cohésion des équipes**

Programme :

Une Code-Retreat est une occasion unique de sortir de sa zone de confort pour les développeurs. Par la répétition d'un même exercice avec des partenaires différents et en effaçant le code à chaque fois, ce format à expérimenter absolument au moins une fois, permet d'essayer des façons différentes de coder, de tester, de faire du design et de collaborer.

Organiser une Code-Retreat, c'est libérer quelques instants les développeurs de la pression quotidienne pour prendre du recul, réfléchir à un sujet précis et progresser.

Une Code-Retreat est un événement pratique d'une journée, qui se concentre sur les principes fondamentaux du développement et de la conception de logiciels. En offrant aux développeurs la possibilité de prendre part à une pratique ciblée, loin de la pression de livrer, le format code-retreat s'avère être un moyen très efficace d'amélioration des compétences.

Le format de la Code-Retreat a été conçu par les développeurs pour les développeurs, sur les principes des Coding Dojo. Une Code-Retreat dure en principe une journée entière. C'est un événement qui met en avant le Test-Driven Development, le pair-programming, le design de code au travers du refactoring, et la recherche de la qualité de code.

Public :

Cet évènement s'adresse principalement aux développeurs.

Pré-requis :

- Avoir envie de découvrir une approche moins traditionnelle et hors des sentiers battus.

Matériel pédagogique :

Le formateur distribuera les supports de la formation au format électronique PDF à la fin de la formation.

[Formations]

Test Driven Development (TDD)

2 jours	**Méthodes pédagogiques :** 40% théorique, 60% de travaux pratiques
Prochaines sessions : retrouvez les dates et le calendrier complet sur notre site www.arolla.fr/training	
Offre inter-entreprise : Petit-déjeuner - Déjeuner - Café et collation	

Objectifs :

Après cette formation TDD, vous saurez utiliser des tests automatisés comme moyen de spécification, de conception et bien sûr de test. Vous saurez développer des tests pour du code existant hérité.

Vous connaîtrez les techniques et outils, tels les doublures et Mocks, pour développer efficacement en Test Driven Design.

Les objectifs ambitieux et de qualité fixés en début de projet résistent rarement aux contraintes du quotidien. Comment maintenir la qualité d'un code, qui tend naturellement à diminuer, si les tentatives d'amélioration se traduisent par des régressions incontrôlées ?

Le développement piloté par les tests (TDD) propose une solution radicale pour reprendre la maîtrise du code et de la conception : les tests systématiques sont écrits avant le code à tester !

Le retour sur investissement peut être important (qualité du code, non régression, évolutivité, maintenabilité). Encore faut-il éviter les écueils d'une mise en œuvre inadaptée de la démarche.

Programme :

- Découverte des principes fondamentaux et de bonnes pratiques du TDD
- Utilisation des Frameworks de test dans une approche TDD
- Utilisation des techniques avancées d'écriture de tests
- Mise en œuvre du TDD en présence de code hérité (legacy)
- Application du TDD dans des contextes particuliers (bases de données, IHM)
- Pratique du refactoring d'un code développé en TDD

A savoir :

- Vous verrez le TDD à l'œuvre au travers d'exemples concrets et d'outils disponibles sur les plates-formes Java et .Net. Une étude de cas réaliste vous permettra d'acquérir les réflexes du TDD, d'aborder les divers problèmes rencontrés par les développeurs en TDD et de mettre en œuvre les bonnes pratiques, des plus simples aux plus élaborées.

- Cette formation pourra aussi être proposée dans d'autres langages comme C++.

Public :

Cette formation s'adresse aux développeurs, mais aussi aux responsables tests, aux chefs de projets et aux responsables qualité.

Pré-requis :

- Pratique de la conception objet
- Pratique du développement avec Java ou C#

Matériel pédagogique :

Le formateur distribuera les supports de la formation au format électronique PDF à la fin de la formation.

[Formations]

Working On Legacy Code

2 jours	**Méthodes pédagogiques :** 30% théorique, 70% pratique
Prochaines sessions : retrouvez les dates et le calendrier complet sur notre site www.arolla.fr/training	
Offre inter-entreprise : Petit-déjeuner - Déjeuner - Café et collation	

Objectifs :

Découvrir les meilleures techniques permettant de :

- **Reprendre le contrôle d'une application devenue au fil du temps sauvage et incompréhensible**
- **La réfectionner dans les règles de l'art et à iso fonctionnalités**
- **Transformer un étalon engourdi et indiscipliné en un pur-sang tout aussi malléable que fougueux, qui vous obéira au doigt et à l'œil.**

Jeune Padawan ! Nous passerons en revue tout un ensemble d'outils et de techniques dignes d'un Grand Maître Jedi du Code Legacy. Ensemble, nous nous entraînerons dur, à travers toute une panoplie d'exercices pratiques qui vous permettront d'intégrer les bons réflexes, de reprendre à la fois confiance en vous et en votre code. L'objectif de cette formation est de vous armer d'un bagage méthodologique qui vous permettra de devenir autonome face à du code Legacy.

Néanmoins ne vous faites pas d'illusion : vous confronter à du code Legacy restera toujours un exercice très difficile ! Cependant, cette tâche ne vous semblera plus jamais insurmontable.

A travers cette formation, nous nous fixerons notamment comme objectifs :

- Rendre testable une application a priori réputée non testable en l'état
- Contrôler des dépendances sur lesquelles on n'a pas forcément la main à l'aide de bouchons
- Savoir par où commencer et avec quelles stratégies
- De l'importance de connaître parfaitement son IDE
- De la nécessité d'automatiser au maximum

A l'issue de cette formation vous maîtriserez les points suivants:

- Définir ce qu'est du code Legacy et l'identifier
- Faire la différence entre un mock et un stub
- Établir la stratégie la plus adaptée pour rendre du code Legacy testable
- Maîtriser son IDE pour éviter des erreurs manuelles

Programme :

Jour 1

- Définir et identifier ce qu'est du code Legacy
- Reconnaître les « code smells »
- Entraînement aux techniques de refactoring en s'appuyant sur un IDE
- Casser les dépendances afin de pouvoir placer un harnais de tests à l'aide des bouchons

Jour 2

La journée entière sera consacrée à des mises en situation, à travers le traitement de plusieurs petits projets legacy qui permettront de mettre en œuvre ce qui a été abordé en première journée.

A savoir :

- La formation est donnée en français.
- Les supports écrits sont pour la plupart en anglais.
- Les exemples de code éventuels sont en Java.

Public :

Cette formation s'adresse à un public de développeurs.

Pré-requis :

- Avoir pratiqué le TDD (Test-Driven Development) et être familier de la notion de Clean Code est un plus.
- Savoir utiliser Git/SVN.
- Bonne pratique de la conception objet.
- Pour les développeurs Java, savoir utiliser Maven.
- Pour les développeurs C#, la connaissance de Resharper et NCrunch est recommandée.
- Les participants doivent avoir un poste de travail avec un environnement de développement (si besoin, nous consulter).
- Git ou SVN.
- Pour les développeurs Java, un IDE similaire à Eclipse avec le plugin infinitest ou JUnit Loop et Eclemma.
- Pour les développeurs C#, Visual Studio avec Resharper et NCrunch.

Matériel pédagogique :

Le formateur distribuera les supports de la formation au format électronique PDF à la fin de la formation.

[Formations]

Behavior Driven Development (BDD)

2 jours	**Méthodes pédagogiques :** 30% théorique, 70% pratique
Prochaines sessions : retrouvez les dates et le calendrier complet sur notre site www.arolla.fr/training	
Offre inter-entreprise : Petit-déjeuner - Déjeuner - Café et collation	

Objectifs :

Découvrir une approche agile permettant d'améliorer la communication, la compréhension des besoins et de réaliser le produit qu'il faut !

Nous verrons comment l'utilisation d'exemples permet d'enrichir les spécifications de manière efficace en construisant un langage commun, partagé et compréhensible de tous.

L'objectif de cette formation est de montrer que l'écriture des spécifications et l'écriture des tests d'acceptation sont des activités complémentaires qui devraient être menées simultanément pour peu qu'elles soient abordées de manière collaborative. Nous verrons comment cela aboutit d'ailleurs à une meilleure compréhension et réalisation des demandes.

A travers plusieurs exercices, nous aborderons notamment différentes manières de :

- Remettre la collaboration au centre du développement
- Rendre nos spécifications « non ambigües »
- Réfléchir à plusieurs pour être le plus « complet » possible
- Éliciter et décomposer des besoins complexes en tâches appréhendables
- Rendre des spécifications exécutables
- Constituer une documentation vivante
- Mettre en place des ateliers de spécification

A l'issue de cette formation vous maîtriserez les points suivants :

- Comprendre les différences entre TDD et BDD
- Mettre en place BDD dans votre organisation
- Écrire des exemples sur des cas complexes
- Challenger des exemples déjà écrits

Programme :

BDD : le pourquoi et le comment

- Les spécifications, un constat d'échec ?
- Origines du BDD
- Collaboration et responsabilité collective

Écriture d'exemples

- C'est quoi un exemple ?
- Critères d'acceptation vs exemples
- Gherkin ou la revanche du Given/When/Then

User Stories et exemples ?

- L'agilité dans tous ses états
- Découpons, découpons ! …mais pas trop !

Automatisation et documentation vivante

- La pyramide de tests
- Gherkin un langage pas comme les autres

Steps

- Table
- Scenario Outline
- Background et Hook
- Revue rapide des différents outils d'automatisation (jBehave, Cucumber, Specflow…)
- Introduction rapide à l'outil Cucumber (pour tout autre outil : jBehave, Specflow, nous consulter…)
- Intégration continue
- Un bien beau rapport d'exécution, des spécifications vérifiées ?!?

NB : selon les profils présents, la partie sur l'automatisation pourra être plus ou moins approfondie.

Public :

Cette formation s'adresse à toute personne impliquée dans le développement logiciel : responsable d'application, chef de projet, maîtrise d'ouvrage, analyste fonctionnel, maîtrise d'œuvre, développeur, testeur, architecte…

Pré-requis :

Aucune connaissance technique n'est requise, cette formation s'adresse aussi bien à des managers, chefs de projets, AMOA, MOA, Business Analysts qu'à des développeurs. Les exemples seront principalement fournis en Java (pour les personnes ayant un background en C#, ou dans un autre langage, nous consulter).

Matériel pédagogique :

Le formateur distribuera les supports de la formation au format électronique PDF à la fin de la formation.

[Formations]

Domain Driven Design (DDD) par Cyrille MARTRAIRE

2 jours	**Méthodes pédagogiques :** 50% théorique, 50% pratique en code et jeux
Prochaines sessions : retrouvez les dates et le calendrier complet sur notre site www.arolla.fr/training	
Offre inter-entreprise : Petit-déjeuner - Déjeuner - Café et collation	

Objectifs :

L'objectif n'est pas de couvrir tout DDD mais de bien comprendre ce qui est important et comment tous ces concepts s'imbriquent entre eux.

Vous souhaitez passer à la vitesse supérieure en design ? Vous vous intéressez à DDD mais ce sujet vous paraît difficile et mystérieux ?

Venez découvrir Domain-Driven Design à Paris !

Lors de cette formation de deux jours, vous apprendrez les points essentiels de DDD par la pratique, grâce à de nombreux exercices entrecoupés de présentations des concepts théoriques.

Domain-Driven Design est une approche complète et approfondie de design pour construire des logiciels qui résolvent des problèmes complexes. DDD apporte des façons de penser pour raisonner efficacement et prendre des décisions éclairées, quel que soit le langage de programmation et l'environnement technologique.

En s'intéressant de très près au domaine métier et en particulier au cœur de métier, DDD vise à aligner le plus littéralement possible le code avec une compréhension approfondie du métier. Cela permet de répondre très rapidement aux évolutions des demandes, et même de libérer des opportunités de nouvelle valeur pour le business.

Programme :

- Aperçu général de Domain-Driven Design
- Comment découvrir un nouveau domaine métier, identifier les sources de savoir-faire et identifier les invariants
- Comprendre la notion d'Ubiquituous Language (langage omniprésent), l'importance du langage, les pièges fréquents et les astuces pour améliorer la communication
- Relation entre DDD et BDD
- Model-Storming pour analyser un métier sous forme de brainstorming collectif
- Expérimenter le design émergent et modéliser itérativement directement en code
- Bien comprendre les Bounded Contexts
- Strategic Design avec Context Mapping et les aspects organisationnels
- Utiliser les patterns tactiques: Value Object, Entity, Domain Service, Hexagonal Architecture
- Aperçu de CQRS et d'Eventual Consistency
- Documentation et Domain-Driven Design

A savoir :

- La formation est donnée en français.
- Les supports écrits sont pour la plupart en anglais.
- Les exemples de code éventuels sont en Java.

Formateur:

Avec plus de 18 ans d'expérience pratique en développement de logiciels, Cyrille Martraire est un coach, un mentor chevronné en conception de logiciels et un praticien internationalement reconnu en DDD.

Public :

Pour permettre les exercices et des échanges de qualité, la formation se fait en petit groupe de 4 à 8 personnes.
Vous êtes développeur logiciel avec une bonne expérience de développement orienté objet et/ou fonctionnel, avec idéalement plusieurs années d'expérience sur des domaines complexes.

Pré-requis :

- Avoir lu le livre d'Eric Evans est un plus, mais non requis.

Matériel pédagogique :

Le formateur distribuera les supports de la formation au format électronique PDF à la fin de la formation.

Cyrille MARTRAIRE est cofondateur technique et CTO d'Arolla.
Il a fondé la communauté Software Crafters Paris. Cyrille Martraire est un praticien internationalement reconnu en DDD. Il anime de nombreuses conférences et formations à ce sujet.

[Formations]

Domain Driven Design (DDD)

2 jours	**Méthodes pédagogiques :** 50% théorique, 50% pratique en code et jeux
Prochaines sessions : retrouvez les dates et le calendrier complet sur notre site www.arolla.fr/training	
Offre inter-entreprise : Petit-déjeuner - Déjeuner - Café et collation	

Objectifs :

L'objectif n'est pas de couvrir tout DDD mais de bien comprendre ce qui est important et comment tous ces concepts s'imbriquent entre eux.

Vous souhaitez passer à la vitesse supérieure en design ? Vous vous intéressez à DDD mais ce sujet vous paraît difficile et mystérieux ?

Venez découvrir Domain-Driven Design à Paris !

Lors de cette formation de deux jours, vous apprendrez les points essentiels de DDD par la pratique, grâce à de nombreux exercices entrecoupés de présentations des concepts théoriques.

Domain-Driven Design est une approche complète et approfondie de design pour construire des logiciels qui résolvent des problèmes complexes. DDD apporte des façons de penser pour raisonner efficacement et prendre des décisions éclairées, quel que soit le langage de programmation et l'environnement technologique.

En s'intéressant de très près au domaine métier et en particulier au cœur de métier, DDD vise à aligner le plus littéralement possible le code avec une compréhension approfondie du métier. Cela permet de répondre très rapidement aux évolutions des demandes, et même de libérer des opportunités de nouvelle valeur pour le business.

Programme :

- Aperçu général de Domain-Driven Design
- Comment découvrir un nouveau domaine métier, identifier les sources de savoir-faire et identifier les invariants
- Comprendre la notion d'Ubiquituous Language (langage omniprésent), l'importance du langage, les pièges fréquents et les astuces pour améliorer la communication
- Relation entre DDD et BDD
- Model-Storming pour analyser un métier sous forme de brainstorming collectif
- Expérimenter le design émergent et modéliser itérativement directement en code
- Bien comprendre les Bounded Contexts
- Strategic Design avec Context Mapping et les aspects organisationnels
- Utiliser les patterns tactiques: Value Object, Entity, Domain Service, Hexagonal Architecture
- Aperçu de CQRS et d'Eventual Consistency
- Documentation et Domain-Driven Design

A savoir :

- La formation est donnée en français.
- Les supports écrits sont pour la plupart en anglais.
- Les exemples de code éventuels sont en Java.

Formateur:

Avec plus de 18 ans d'expérience pratique en développement de logiciels, Cyrille Martraire est un coach, un mentor chevronné en conception de logiciels et un praticien internationalement reconnu en DDD.

Public :

Pour permettre les exercices et des échanges de qualité, la formation se fait en petit groupe de 4 à 8 personnes.
Vous êtes développeur logiciel avec une bonne expérience de développement orienté objet et/ou fonctionnel, avec idéalement plusieurs années d'expérience sur des domaines complexes.

Pré-requis :

- Avoir lu le livre d'Eric Evans est un plus, mais non requis.

Matériel pédagogique :

Le formateur distribuera les supports de la formation au format électronique PDF à la fin de la formation.

[Formations]

Crafting Front-End Code

2 jours	**Méthodes pédagogiques :** 30% théorique, 70% pratique en code et jeux
Prochaines sessions : retrouvez les dates et le calendrier complet sur notre site www.arolla.fr/training	
Offre inter-entreprise : Petit-déjeuner - Déjeuner - Café et collation	

Objectifs :

Cette formation s'appuie sur deux technologies, React et Redux, afin de vous donner les clés pour rentrer dans le sujet et délivrer rapidement de la valeur.

Toutefois, le paysage technologique du front-end étant encore en pleine évolution, cette formation a pour ambition de vous donner à voir au-delà des technologies du moment et s'attachera à dégager les problématiques qui sous-tendent le développement front-end : techniques de composition, de communication entre composants, de gestion de l'état applicatif, de standardisation, voire de coexistence des technologies.

Tout au long de cette formation, un accent particulier sera mis sur la testabilité, la structuration et la maintenabilité, car, nous le verrons, toutes les pratiques du craftsmanship s'appliquent, à commencer par le TDD et le BDD. Cette formation se veut donc théorique et pratique.

Programme :

JavaScript & TypeScript 101 :

- Objets, prototypage, fonctions, scope, asynchronisme (JavaScript) et typage structurel (TypeScript)

Construction progressive d'une application, permettant d'aborder les notions suivantes :

- React : Virtual DOM, Elements, JSX, (Stateless) Components, Hooks, Reconciliation, (Un)controlled Components, Render props, Higher-Order Components, Context API
- Redux : Event sourcing, Immutability, Middlewares (orchestration, asynchronisme), Normalisation
- Testabilité : tests unitaires des différents stéréotypes, dont les composants (émulation du DOM, snapshots), tests d'intégration : profondeur de test, tests end-to-end (page model et BDD)

A savoir :

- La formation est donnée en français, mais les supports de formation sont en anglais.
- Les exemples de code sont en TypeScript, langage qui offre un typage statique et transpile en JavaScript. Il est possible de suivre la formation en travaillant exclusivement en JavaScript.
- Durant la formation, les apprentis seront incités à travailler en pair programming, sans caractère obligatoire bien évidemment.

Public :

Cette formation s'adresse à des développeurs confirmés souhaitant se former au développement front-end.

Nombre de participants :

Jusqu'à 8 personnes par session.

Pré-requis :

- Sensibilité aux pratiques du Craftsmanship, stratégies de tests, notions de programmation fonctionnelle.

Matériel pédagogique :

Repository GitHub du formateur et support de présentation au format PDF.

Formateur :

Formation assurée par Mathieu Eveillard, développeur JavaScript et coach craft. Son expertise porte sur les Single Page Applications, la programmation fonctionnelle et réactive ainsi que React et Redux.

https://mathieueveillard.github.io/

[Formations]

Programmation Fonctionnelle en F#

2 jours (version condensée)	**Méthodes pédagogiques :** 30% de théorie et 70% de pratique
3 jours (version complète)	**Méthodes pédagogiques :** 30% de théorie et 70% de pratique
Prochaines sessions : retrouvez les dates et le calendrier complet sur notre site www.arolla.fr/training	
Offre inter-entreprise : Petit-déjeuner - Déjeuner - Café et collation	

Objectifs :

Cette présentation rassemble tout ce qu'il faut savoir pour aborder sereinement des problématiques d'architecture en entreprise, sur des systèmes qui peuvent être complexes. Elle est directement issue de l'état de l'art validé par nos retours d'expérience depuis des années chez nos clients.

Présentée par un expert avec une forte expérience personnelle en architecture dans des contextes différents, cette formation mixe les présentations théoriques avec des exercices qui permettent aux stagiaires de s'impliquer et de valider leur compréhension.

L'architecture en contexte d'entreprise recouvre des aspects fortement techniques, ainsi que des enjeux humains importants ; c'est la raison pour laquelle cette formation couvre ces deux facettes afin d'offrir une vision complète du sujet.

Programme :

Introduction

- F# et l'écosystème .NET
- Installation de F#

Prise en main du langage

- Structuration d'un projet F#
- Les différents fichiers (.fs, .fsx, .fsi)
- Le REPL
- Déclarer des valeurs
- Déclarer des fonctions
- L'inférence de type
- Les opérateurs

Vue d'ensemble des structures de données

- List, Array, Sequences, Map, Set
- Les types algébriques
- Tuple
- Record type
- Discriminated Union
- Les types optionnels
- Les unités de mesure

Illustration des concepts liés à la programmation fonctionnelle avec F#

- L'immutabilité
- Les expressions
- Le Pattern Matching
- Les active patterns
- Le Currying
- L'application partielle
- La composition
- Les fonctions de haut niveau

Public :

Cette formation s'adresse aux développeurs expérimentés, team leads, responsables d'architectes, mais aussi aux architectes confirmés ou en devenir .

Matériel pédagogique :

Le formateur distribuera les supports de la formation au format électronique PDF à la fin de la formation.

[Formations]

Programmation fonctionnelle en Java

2 jours	**Méthodes pédagogiques :** 50% théorique, 50% pratique (exercices à base de code)
Prochaines sessions : retrouvez les dates et le calendrier complet sur notre site www.arolla.fr/training	
Offre inter-entreprise : Petit-déjeuner - Déjeuner - Café et collation	

Objectifs :

L'objectif est d'apprendre le paradigme de la programmation fonctionnelle, de comprendre les concepts, avantages liés à cette approche et de la mettre en oeuvre dans les développements en langage Java, notamment en tirant partie des évolutions du langage introduites à partir de la version 8.

Lors de cette formation de deux jours, vous apprendrez les points essentiels de la programmation fonctionnelle par la pratique, grâce à de nombreux exercices entrecoupés de présentation des concepts théoriques.

La programmation fonctionnelle est un paradigme de programmation qui utilise les fonctions comme brique principale de construction de logiciel.

Elle permet de maîtriser la complexité accidentelle dans la réalisation de logiciel, de réduire la duplication de code, réduire les sources de bug, de rendre le code plus facilement testable et exécutable même de manière parallèle.

Pour pouvoir tirer parti de ce paradigme, il faut connaître les bonnes pratiques clés pour le mettre en oeuvre surtout quand on a l'habitude de mettre en oeuvre d'autres paradigmes (impératifs ou orienté-objet, etc…).

Programme :

Qu'est que la programmation fonctionnelle ?

- Les bénéfices de la programmation fonctionnelle

Les problèmes avec les effets de bord

- Les différences entre les fonctions pures et impures. Comment la transparence référentielle rend les programmes plus sûrs. Raisonnement sur les programmes avec le modèle de substitution
- Tirer le meilleur parti de l'abstraction

Utilisation des fonctions en Java

- Représentation des fonctions en Java
- Utilisation des lambdas et des méthodes références
- Utilisation des fonctions d'ordre supérieur
- Utilisation de la curryfication

Rendre Java plus fonctionnel

- Rendre les structures de contrôle standard fonctionnelles
- Abstraction des structures de contrôle
- Abstraction des itérations
- Utiliser les bon types

Récursion, corécursion et mémorisation

- Récursion et corécursion
- Fonctions récursives
- Composer un grand nombre de fonctions
- Mémorisation automatique

Le traitement des données avec des listes

- Classement des structures de données en programmation fonctionnelle
- Utilisation de l'omniprésente simple liste chaînée
- Comprendre l'importance de l'immutabilité
- Traitement des lists avec récursion et fonctions

La gestion des données optionnelles.

- La référence null ou « l'erreur qui vaut un milliard »
- Les alternatives à la référence null
- Développement d'un type de donnée Option pour les données optionnelles
- Appliquer des fonctions aux valeurs optionnelles
- Composition de valeurs optionnelles
- Les cas d'utilisation d'Option

Mais aussi:

- Gestion des erreurs et des exceptions
- Support avancé des listes
- Travailler de manière paresseuse
- Plus de données traitées avec les arbres.
- Supporter les changements d'états de façon fonctionnelle
- Entrées / sorties fonctionnelles
- Les nouveautés ajoutées en Java 8

A savoir :

- La formation est donnée en français.
- Les supports sont écrits en français.
- Les exercices et exemples sont en Java 8 ou plus.

Public :

Vous êtes développeur logiciel expérimenté, bien-sûr, en Java !

Pré-requis :

- Connaître le langage Java.

Matériel pédagogique :

Le formateur distribuera les supports de la formation au format électronique PDF à la fin de la formation.

[Formations]

Microservices avec Domain-Driven Design

2 jours	**Méthodes pédagogiques :** 60% théorique, 40% pratique à travers des ateliers de mise en pratique
Prochaines sessions : retrouvez les dates et le calendrier complet sur notre site www.arolla.fr/training	
Offre inter-entreprise : Petit-déjeuner - Déjeuner - Café et collation	

Objectifs :

- **Définir une architecture basée sur les microservices**
- **Maîtriser les concepts essentiels du Domain-Driven Design (DDD)**
- **Apprendre à modéliser et à définir le périmètre d'un microservice**
- **Identifier les problématiques et les patterns de résolution d'une architecture microservices**
- **Définir une trajectoire de migration d'un existant vers une approche microservices**

L'architecture microservice est une approche attractive pour construire et faire évoluer des systèmes à grande échelle, avec un grand nombre d'utilisateurs ou avec un grand nombre d'équipes de développeurs qui doivent pouvoir travailler en relative autonomie pour livrer des fonctionnalités efficacement et indépendamment.

En matière de microservices, le focus est souvent sur les technologies, alors que les points clés sont la définition pertinente des contours des services. Ce n'est pas facile et c'est précisément pour cela que l'approche Domain-Driven Design est essentielle pour guider le découpage. Il s'agit aussi de penser en terme de système en soignant tout ce qui est nécessaire entre les services.

Cette formation couvre à la fois la théorie et les applications dans une présentation unifiée, avec des exercices pratiques. Elle présente les concepts essentiels de DDD, les techniques clés d'architecture ainsi que les principales technologies nécessaires pour les mettre en oeuvre. Et puisque cette mise en oeuvre commence le plus souvent sur des systèmes existants, cette formation décrit aussi des trajectoires de migration et de cohabitation avec le legacy.

Cette formation s'appuie sur notre expertise théorique sur le sujet et sur nos expériences d'accompagnement de nos clients sur ces architectures microservices. En particulier, nous décrivons aussi des modes d'échec classiques que nous avons observés.

Programme :

Faire le point sur les différentes définitions et enjeux des microservices

- API et approches API-first
- Autres façons de regarder les microservices : « damage control »: size, time to replace, decoupling, scalabilité, SOLID
- Polyglot persistence, polyglot services

Comprendre l'apport de Domain-Driven Design pour aider à définir les frontières des services

- Comprendre et identifier les Bounded Contexts
- Découvrir des notions de Context Mapping
- Réaliser les conséquences en terme de duplication (DRY vs Coupling) et de cohérence à terme (Eventual Consistency)

Comprendre une architecture microservices Architecture

- Comprendre la différence entre Choreography et Orchestration
- Intégration Asynchrone avec Event-Driven Architecture, messaging et patterns d'intégration EIP
- Intégration Synchrone et les conséquences
- Réaliser l'importance de la robustesse des contrats entre services, leur gouvernance pour les enjeux de compatibilité, les heuristiques et pièges à éviter
- Compromis Autonomy vs. Authority
- CQRS, Backend for Front-End (UI-specific API vs service API), mashup techniques

Penser "Design for Failure"

- Penser en terme de zones d'échec (Failure Areas)
- Circuit breakers for graceful degradation
- Découvrir les approches de test : compile time (Consumer-Driven Contracts) or runtime (Symian Army)
- When data get out of sync : reconciliation, replay, reload, et l'importance de l'idempotence et du polling comme solutions
- Monitoring : guidelines, outils, Alerting & Distributed tracing

Implémenter un service

- Concevoir une API, au-delà de la capacité "d'exposer les entités en REST"
- Persistence privative comme source de vérité (Sources of Truth), pattern Data Pump
- L'importance d'être Stateless, pour la fault-tolerance, le Cloud et l'elasticity
- Architecture Hexagonale
- Event-Sourcing vs Event-Driven
- Découvrir des patterns spécifiques pour le Cloud, la notions de Serverless & de Function as a Service
- Microservice Chassis: Spring Boot, Dropwizard
- Horizontal Scaling, Load balancing and intelligent routing
- Configuration services, Registration and Discovery
- Principes de sécurité

Migration Legacy

- Requalifier un existant legacy en "Macroservices"
- Extension d'un existant par des nouveaux services dans une approche de fédération
- Le pattern clé Strangler Application et autres patterns legacy utiles : Legacy Read Model, Double Feeding, Services Aggregation, Change Data Capture
- Les enjeux des périmètres transactionnels historiques : Save on Out of Focus vs. Distributed Transaction vs. Large microservices

Introduire une architecture microservices et les principaux pièges à éviter

- A quel moment et comment adopter les microservices
- Un changement de culture au-delà d'un changement technique
- Alternatives aux microservices : modularité, Discipline, approche microservice-Ready, et les limites associées
- Compétences et maturité pré-requises

Public :

Cette formation s'adresse aux développeurs expérimentés, aux architectes, team leads, référents et coachs techniques.

Pré-requis :

- Bonne connaissance d'au moins un langage de programmation.
- Les participants doivent avoir un poste de travail avec un environnement de développement.
- Avoir déjà une expérience dans la conception logicielle dans le cadre de projets d'entreprise.

Matériel pédagogique :

Le formateur distribuera les supports de la formation au format électronique PDF à la fin de la formation.

Living Documentation par l'auteur

1 jour	**Méthodes pédagogiques :** 60% théorique – 40% pratique à travers des ateliers de mise en pratique
Prochaines sessions : retrouvez les dates et le calendrier complet sur notre site www.arolla.fr/training	
Offre inter-entreprise : Petit-déjeuner - Déjeuner - Café et collation	

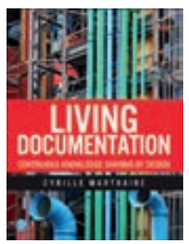
Objectifs :

- **Comprendre l'approche Living Documentation**
- **Comprendre en quoi cette approche bouleverse la gestion des connaissances traditionnelles**
- **Découvrir les pratiques essentielles en théorie et par une mise en pratique**
- **Découvrir que la documentation est aussi un outil de feedback rapide**
- **Apprendre des moyens d'introduire cette approche dans votre environnement**

Le transfert de connaissance est un enjeu majeur en développement logiciel. Cet enjeu est particulièrement aigu dans le long terme et avec du turnover dans les équipes. Pourtant, la documentation est souvent négligée, ou représente une quantité de travail importante. Et surtout, personne n'aime passer du temps à créer une documentation qui devient immédiatement obsolète !

L'approche Living Documentation (Documentation Vivante) représente une solution à la fois efficace et attractive, qui permet de maximiser le transfert et la conservation du savoir sans pour autant devenir une taxe élevée pour les membres de l'équipe. Surtout, c'est une approche qui permet l'accélération des rythmes de développement et de changements visés en Continuous Delivery, DevOps et toutes les approches agiles.

Si vous préférez coder plutôt qu'écrire de la documentation, vous aimerez la Living Documentation !



Programme :

Comprendre que la plupart des connaissances sont déjà représentées quelque part dans les artefacts, même sous une forme implicite.

- Réaliser que se parler et travailler ensemble peut être une forme de documentation suffisante, et en connaître les limites
- Apprendre à augmenter le code pour le rendre plus explicite pour une utilisation ultérieure par des humains et outils, par exemple pour baliser un Guided Tour au travers d'un système
- S'entraîner à faire la différence entre les connaissances stables ou volatiles, voir en quoi cela informe sur la façon dont on les manipule, et comment créer des documents Evergreen à zéro maintenance
- Apprendre à reconnaître les Shameful Comments qui suggèrent des améliorations du code
- Apprendre à visualiser la structure du design en générant un Living Diagram directement à partir du code, augmenté si nécessaire
- Découvrir comment un simple nuage de mots peut indiquer les défauts de votre modèle métier
- Apprendre à générer un Glossaire Vivant directement à partir du code
- Se familiariser avec les Architectural Decision Records pour aider à prendre de meilleures décisions, et en garder trace
- Réaliser que les Design Patterns aident à documenter des décisions de design très efficacement
- Comprendre les techniques fondamentales pour répondre à vos besoin spécifiques, et voir les pièges principaux à éviter
- Découvrir que la difficulté à mettre en oeuvre une documentation vivante est en soi un retour utile pour améliorer le système
- Apprendre à introduire le sujet de la documentation vivante dans votre projet et avec vos collègues

A savoir :

- La formation par l'auteur du livre de référence.
- Cette approche est décrite en détail dans le livre « Living Documentation » (ISBN-13: 978-0134689326) annoncé chez Addison-Wesley Professionals, écrit par Cyrille Martraire.
- Et pour démarrer dans de bonnes conditions, Arolla propose en exclusivité la formation par l'auteur. Même si vous n'avez pas lu le livre, au travers de présentations théoriques et d'exemples "les mains dans le code" vous apprendrez les points essentiels et les travers à éviter, et vous comprendrez comment optimiser votre approche de la documentation dans un projet en changement permanent.

Public :

Vous êtes développeur, architecte, team lead, testeur sachant coder ou bien coach ? Cette formation est pour vous !

Pré-requis :

- Tous les exemples se feront en Java (nécessite un poste de travail avec un IDE Java avec Maven installé), mais sont facilement extrapolables à d'autres univers, en particulier .Net.
- Avoir un minimum d'expérience dans la conception logicielle dans le cadre de projets d'entreprise.

Matériel pédagogique :

Cyrille Martraire distribuera les supports de la formation au format électronique PDF à la fin de la formation.

Pragmatic Architecture

2 jours	**Méthodes pédagogiques :** 60% théorique – 40% pratique

Prochaines sessions : retrouvez les dates et le calendrier complet sur notre site www.arolla.fr/training

Offre inter-entreprise : Petit-déjeuner - Déjeuner - Café et collation

Objectifs :

Au cours de cette formation vous allez :

- **Connaître les principaux patterns et styles architecturaux ainsi que l'état de l'art**
- **Concevoir une architecture pragmatique, la documenter et la présenter**
- **Résoudre les principales problématiques d'une architecture en production**
- **Identifier les couplages et les critères de modularité**
- **Découpler les composants et gérer la scalabilité et la disponibilité**
- **Avoir une vision globale du système logiciel**

Cette présentation rassemble tout ce qu'il faut savoir pour aborder sereinement des problématiques d'architecture en entreprise, sur des systèmes qui peuvent être complexes. Elle est directement issue de l'état de l'art validé par nos retours d'expérience depuis des années chez nos clients.

Présentée par un expert avec une forte expérience personnelle en architecture dans des contextes différents, cette formation mixe les présentations théoriques avec des exercices qui permettent aux stagiaires de s'impliquer et de valider leur compréhension.

L'architecture en contexte d'entreprise recouvre des aspects fortement techniques, ainsi que des enjeux humains importants ; c'est la raison pour laquelle cette formation couvre ces deux facettes afin d'offrir une vision complète du sujet.

Programme :

Introduction à l'Architecture

- Les multiples définitions de l'architecture technique : une question de stabilité, grande échelle, savoir partagé, décisions irréversibles
- Architecture comme moyen d'alignement entre les objectifs de l'entreprise et ses systèmes d'information
- Architecture comme styles d'implémentations
- Architecture globale vs. locale, Umbrella diagrams

Overview de styles d'architecture

- Architecture Hexagonale
- Event-Driven Architecture
- Pipeline
- Microservices
- Entreprise Integration Patterns (EIP, Gregor Hohpe), Integration styles
- PoEAA (Fowler)
- API design
- Architecture Émergente

Technical Architecture

- Configuration, Discovery
- Asynchronous messaging
- Service security (A&A, OAuth2, attacks/defenses)
- Data consistency: Golden Source, replication, autonomy vs. authority
- Fault-tolerance, Load balancing
- Stateless vs statefull, consequences, stickiness & affinité, Managing sessions
- Logging & tracing guidelines
- Monitoring: guidelines, tools, alerting
- Integration in practice: reconciliation, data migration
- Test-Driven Architecture, métriques, static analysis

Documentation d'Architecture

- L'importance de la communication en architecture, techniques et astuces
- Living Documentation appliquée à l'architecture : Evergreen Overview Document, Internal Documentation avec annotations et conventions de codage
- Living Diagrams générés à chaque build

Legacy migration patterns

- Extension by Federated Service
- Strangler Application
- Legacy Read Model

Architecture & DDD

- Aperçu des Bounded Contexts et du Context Mapping
- Aperçu des Large-Scale Structures de DDD

Public :

Cette formation s'adresse aux développeurs expérimentés, team leads, responsables d'architectes, mais aussi aux architectes confirmés ou en devenir.

Pré-requis :

- Connaissance de différents langages de programmation avec des bonnes connaissances en design objet. Avoir déjà une expérience dans le développement de logiciel.
- Les participants doivent avoir un poste de travail avec un environnement de développement.

Matériel pédagogique :

Le formateur distribuera les supports de la formation au format électronique PDF à la fin de la formation.

Premiers pas dans Azure

3 jours	**Méthodes pédagogiques :** 50% théorique, 50% pratique en code et jeux
Prochaines sessions : retrouvez les dates et le calendrier complet sur notre site www.arolla.fr/training	
Offre inter-entreprise : Petit-déjeuner - Déjeuner - Café et collation	

Objectifs :

Utiliser plus efficacement les principaux services Azure, les bonnes pratiques et exploiter au mieux cette infrastructure en fonction de vos besoins.

Le cloud permet de mettre en place de la haute disponibilité, de la scalabilité, de la résilience, tout cela en maîtrisant les coûts.

Cette promesse, bien qu'atteignable, nécessite de prendre en compte de nombreux aspects de l'architecture et du développement, ainsi que de choisir et maîtriser les services utilisés. Nous verrons dans cette formation quels patterns sont à privilégier, et comment concevoir dans le cloud.

Programme :

Azure Web App
- Le sizing
- Les slots
- Le swap

Traffic manager
- Load balancer au niveau DNS
- Sa configuration
- Ses différents modes de fonctionnement

Cache Redis Azure
- Pourquoi un cache distribué dans une architecture cloud ?
- Les performances
- Le sizing
- Les types de structures
- Les pièges à éviter
- Cache aside pattern

Cosmos DB
- Azure service bus
- Azure Functions
- Les performances

Azure functions Service Fabric

A première vue, le cloud est plein de promesses. Sur la disponibilité, la tolérance à la panne, la scalabilité, la résilience tout en garantissant une maîtrise des coûts et une optimisation.

Cependant, pour y arriver, les pré-requis techniques sont importants et l'architecture doit avoir été pensée correctement en amont.

La plateforme offrant beaucoup de services offrant eux-mêmes beaucoup de possibilités, nous ne verrons en formation que les plus récurrents, ceux qui ont une place indiscutable dans les architectures Cloud Native.

Le cloud permet d'adapter ses ressources à ses besoins (Scaling horizontal & vertical). Cependant ses mécanismes doivent être utilisés à bon escient et ils ne substituent pas à une démarche d'architecture moderne, sous peine de voir exploser sa facture.

Nous verrons donc comment choisir les services Cloud nous permettant de satisfaire les besoins émergents des architectures modernes et des nouveaux usages.

Public :

Les développeurs ayant de bonnes connaissances en développement logiciel.

Pré-requis :

- Les exemples seront donnés en C#.
- Les participants doivent avoir un poste de travail avec un environnement de développement.

[Offres de conseil]

Coaching Embarqué

Lieu : chez le client.

Durée : de 3 jours consécutifs à 3 semaines, en jours consécutifs ou en enveloppe de jours à planifier selon les besoins.

Le plus grand levier pour élever la culture et les compétences d'une équipe est le casting de ses membres. Quand nos consultants rejoignent vos équipes, au-delà du fait qu' ils apportent leur capacité de travail, ils et elles amènent des compétences de Software Craft ainsi que l'enthousiasme pour les diffuser auprès de leurs collègues. C'est une façon efficace d'amener des pratiques vertueuses, par l'exemple, dans la durée.

- Montrer l'exemple dans le travail au quotidien, dans une posture de co-équipier et non de coach
- Diffusion de compétences en pair-programming ou Mob programming
- Diffusion de compétences lors du travail collectif, lors des revues de code et dans les discussions entre collègues
- Amorçage en douceur des pratiques Clean Code, TDD, BDD, Legacy, par l'exemple et dans l'environnement de l'équipe

Un audit constructif

Lieu : chez le client.

Durée : de 3 jours consécutifs à 3 semaines, en jours consécutifs ou en enveloppe de jours à planifier selon les besoins.

Un audit n'a pas vocation à être punitif, mais objectif et constructif. Il s'agit d'inviter des experts à évaluer la situation afin de détecter les points faibles à améliorer et les points forts à préserver et à soutenir.
Nous considérons votre système dans son ensemble, avec les logiciels et les personnes qui les entourent, des techniques aux pratiques et à la culture, pour une évaluation cohérente.

La restitution peut se faire en continu de façon informelle, avec une présentation interactive au terme, suivie si besoin de la rédaction d'un livrable.

- Recueillir un avis d'expert avant de vous engager dans une décision importante, technique ou concernant l'organisation des développements
- Faire le point sur votre patrimoine logiciel, les points forts et les points d'amélioration
- Évaluer la qualité des livraisons d'un fournisseur
- Identifier des opportunités d'amélioration dans vos processus de développement

Stratégie de test

Lieu : chez le client.

Durée : de 3 jours consécutifs à 3 semaines, en jours consécutifs ou en enveloppe de jours à planifier selon les besoins.

Une stratégie de test est un raisonnement sur les compromis à trouver pour optimiser la qualité au prix d'efforts limités. Par l'analyse des risques, de leurs probabilités et de leurs conséquences, et ce de façon différenciée selon les zones du système (neuf ou ancien, par domaine métier, front ou back etc.) il s'agit de définir des objectifs de priorisation par type de test : manuels ou automatisés, par granularité de test et par type de technologie de tests.

Cette stratégie doit, en outre, être adaptée au contexte interne, dans les technologies et les pratiques déjà présentes ainsi qu'à la maturité des équipes concernées.

- Définir les principaux enjeux de qualité en terme de risques, impacts et probabilités selon les composants de votre système
- Établir une stratégie pour optimiser l'allocation des efforts de tests selon les enjeux identifiés
- Différencier la stratégie selon les types de risques, en mixant les différentes approches de tests automatisés ou humains ainsi que d'autres approches
- Proposer des techniques concrètes pour la mise en œuvre tactique de la stratégie avec, si nécessaire, la réalisation de pilotes.

[Offres de conseil]

Tester l'intestable

Lieu : chez le client.

Durée : de 3 jours consécutifs à 3 semaines, en jours consécutifs ou en enveloppe de jours à planifier selon les besoins.

La détermination ne suffit pas toujours face à un obstacle de taille. C'est le moment de faire appel à des spécialistes pour dégager la voie et baliser la suite, ou pour proposer un contournement le cas échéant. C'est notre rôle de vous apporter expertise et persévérance, par exemple lors d'interventions de type "coup de poing".
Le livrable est une démonstration par l'exemple de l'approche pour rendre testable le composant considéré.

- Identifier les meilleures opportunités pour attaquer un composant non testable a priori
- Définir et mettre en œuvre par l'exemple les jalons jusqu'à réussir à tester efficacement l'incrément de fonctionnalité désirée
- Utiliser cette forme d'intervention en mode «coup de poing » dans un but pédagogique pour toute l'équipe
- Faire la démonstration, symbolique et en pratique, de la volonté du management de consacrer des moyens pour s'attaquer aux problèmes du code legacy

[Offres de conseil]

Faire converger les perspectives

Lieu : chez le client.

Durée : de 3 jours consécutifs à 3 semaines, en jours consécutifs ou en enveloppe de jours à planifier selon les besoins.

Avoir différents points de vue est une richesse qui peut entraîner une paralysie lorsque le consensus est difficile à obtenir. Un regard extérieur, légitime et détaché des enjeux qui recherche une solution médiane, acceptable par tous vous aidera à débloquer cette situation.

- Collecter les différentes perceptions des problèmes et enjeux avec les personnes concernées
- Identifier l'intersection de ce qui est acceptable par tous, en cohérence avec l'expertise et l'expérience de nos consultants
- Converger vers un consensus légitime, accepté par tous, afin de mettre un terme aux discussions et passer enfin à l'exécution

[Offres de conseil]

Aller vers des micro-services à partir d'un monolithe legacy

Lieu : chez le client.

Durée : de 3 jours consécutifs à 3 semaines, en jours consécutifs ou en enveloppe de jours à planifier selon les besoins.

Grâce à nos nombreux clients nous avons une large expérience des découpages en microservices. A partir de la cartographie consensuelle préalable des domaines métiers, nous définissons les meilleurs angles d'attaque pour une transformation progressive à partir d'un système legacy étendu. Vous profiterez de notre expérience de terrain et de notre expertise sur les outils essentiels de Domain-Driven Design aux migrations en natif cloud, en passant par les stratégies legacy et de tests associés.

Notre livrable est une cartographie consensuelle des principaux domaines fonctionnels de votre activité, ainsi que la doctrine adaptée pour guider la migration progressive pas à pas lors des développements à venir.

- Établir une cartographie des sous-domaines de votre métier, qui sont autant de candidats pour définir les contours des futurs services
- A partir de l'état de l'existant, proposer un mode opératoire concret à mettre en œuvre lors des prochains projets transformants pour se rapprocher de la cible, à savoir l'alignement du système sur les contours du métier
- Identification des opportunités et des contraintes majeures pour une cohabitation heureuse entre des nouveaux services et des composants legacy, en utilisant les patterns adaptés et notre expérience d'autres clients

[Offres de conseil]

Définir les tracés directeurs

Lieu : chez le client.

Durée : de 3 jours consécutifs à 3 semaines, en jours consécutifs ou en enveloppe de jours à planifier selon les besoins.

Plus l'enjeu est élevé moins le doute est supportable. Migrer vers une nouvelle architecture ou une nouvelle façon de penser l'architecture est une situation unique pour chacun de nos clients, mais c'est une situation habituelle pour nous. Nous vous aidons à prendre des décisions motivées par des connaissances puis à baliser le chemin.

- Aider les équipes à formaliser leurs intentions, par exemple avec une nouvelle architecture
- Apporter un avis d'expert extérieur et légitime, issu des retours d'expériences de clients comparables, pour optimiser les propositions
- Définir de façon consensuelle les jalons importants (sur papier, en code ou par des objectifs de métriques), tout en évitant les pièges les plus fréquents

[Offres de conseil]

Faire grandir les équipes, renforcer les champions internes

Lieu : chez le client.

Durée : de 3 jours consécutifs à 3 semaines, en jours consécutifs ou en enveloppe de jours à planifier selon les besoins.

L'entraînement sur des katas consiste à pratiquer des gestes particuliers de façon délibérée pour progresser, parfois avec des contraintes non-ordinaires, avec les conseils d'un coach expérimenté. Nos coachs en pratiques d'ingénierie agile (Software Craftsmanship) accompagnent la montée en compétences et aident vos champions à prendre le relai.

- Coaching d'équipes autour des techniques d'ingénierie agile
- Mise en place et amorçage de cycles de Katas internes
- Animation de coding dojos et de sessions de montée en compétence
- Coaching ciblé des champions internes qui doivent prendre le relai à terme

[Offres de conseil]

Booster une initiative, valoriser les savoirs & les expériences

Lieu : chez le client.

Durée : de 3 jours consécutifs à 3 semaines, en jours consécutifs ou en enveloppe de jours à planifier selon les besoins.

Un événement interne, du kata hebdomadaire en équipe à la conférence à échelle mondiale, est l'occasion de créer des liens et du vécu partagé, au-delà du partage de connaissances. Nous organisons tout cela pour nos clients.

- Mise en place d'évènements de code de type Coding Game ou Hackathon
- Création de Coding Game sur-mesure autour de vos besoins et enjeux spécifiques
- Création d'une conférence interne à destination des équipes de développement pour soutenir une initiative
- Animation de médias internes ou externes de partage d'expérience : interviews de développeurs, interviews vidéo, contenu rédactionnel sur mesure
- Création de formats innovants pour accompagner les CXO dans leur compréhension des enjeux du développement logiciel

[Offres de conseil]

Rendre lisible la complexité

Lieu : chez le client.

Durée : de 3 jours consécutifs à 3 semaines, en jours consécutifs ou en enveloppe de jours à planifier selon les besoins.

Savoir regarder est une compétence qui fait la différence. Notre savoir-faire et notre culture en systèmes logiciels constituent une grille de déchiffrage pour regarder les problèmes de design, de technique, de culture technique sous le bon angle.

- Modéliser des domaines métiers complexes pour la première fois
- Modéliser ou re-modéliser pour de meilleures performances ou une meilleure pertinence par rapport aux attentes du métier
- Ré-implémenter une grande portion de code laborieux en un modèle métier élégant et concis
- Introduire une petite dose de sophistication pour simplifier grandement un problème

[Offres de conseil]

Création d'outils spécifiques

Lieu : chez le client.

Durée : de 3 jours consécutifs à 3 semaines, en jours consécutifs ou en enveloppe de jours à planifier selon les besoins.

Nous avons une expérience des métriques et des techniques pour rendre plus visibles les évolutions d'un système et des comportements sur ce système. Au-delà des métriques et solutions standards, nous pouvons concevoir des outils et des modes de mesure spécifiques pour vous aider à piloter, en évitant les pièges classiques des indicateurs.

- Création de plugin sur-mesure dans Sonar, Jenkins, Cucumber ou autre outil, par exemple pour aider au suivi de points d'attention spécifiques à votre contexte
- Création d'outils de tests spécialisés pour des contextes difficiles ou non standards, en local ou dans le Cloud
- De façon générale, tout développement sur mesure d'outils ou d'application

[Offres de conseil]

Accompagner votre croissance au-delà des changements d'échelle

Lieu : chez le client.

Durée : de 3 jours consécutifs à 3 semaines, en jours consécutifs ou en enveloppe de jours à planifier selon les besoins.

Changer d'échelle change tout, en nature et en performance. Quand la NASA a construit des hangars géants pour protéger les fusées des intempéries, un micro-climat imprévu est apparu dans ces hangars gigantesques. Nous vous accompagnons pour organiser des équipes efficacement, pour vraiment livrer deux fois plus en doublant le nombre d'équipes. Cela implique de faire grandir les talents existants, adapter les pratiques d'architecture, tout en étant attentif à l'évolution de la culture.

- Identifier les opportunités pour obtenir des flux de travail indépendants afin de réduire les coûts de coordination
- Encourager l'intelligence locale pour gagner en qualité et en vitesse
- Mettre en place des initiatives légères de gouvernance pour assurer la cohérence globale au moindre coût
- Analyser les flux de demandes de nouvelles fonctionnalités, pour suggérer des possibilités d'alignement avec les équipes
- Analyser les zones métier candidates à être des zones d'évolution indépendantes (Bounded Contexts)
- Analyser les avantages et inconvénients des Feature Teams vs. Component Teams dans votre propre contexte
- Recommandations sur l'organisation des équipes de développement et des interlocuteurs métier

[Offres de conseil]

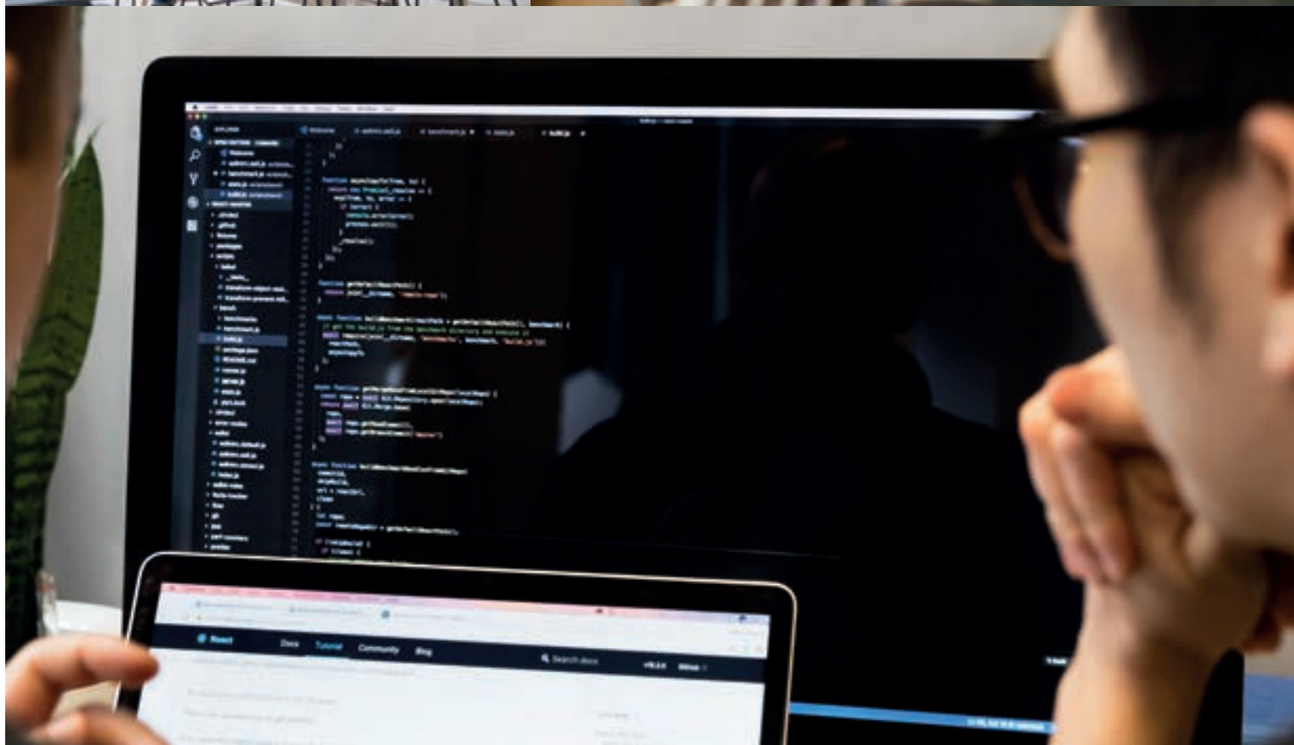
Accompagner une transformation d'envergure

Lieu : chez le client.

Durée : de 3 jours consécutifs à 3 semaines, en jours consécutifs ou en enveloppe de jours à planifier selon les besoins.

Accompagner une transformation, c'est aider chaque équipe à muscler ses savoir-faire et habitudes technique ainsi qu'à se ré-organiser pour réfléchir et interagir différemment, par exemple avec des équipes qui concentrent toutes les compétences essentielles pour avancer au plus vite. Il s'agit en particulier de ré-aligner l'organisation des équipes sur les flux de besoins extérieurs. C'est enfin un volet technique avec les pratiques d'architecture nouvelles et de transition avec l'existant, à la recherche de l'équilibre idéal.

- Aider les équipes à s'engager progressivement dans une plus grande autonomie
- Former des équipes poly-compétentes (Whole Teams) sur les compétences clés dans leur contexte propre, et ce dans la limite de la taille maximale d'une équipe
- Encourager les équipes à prendre en charge l'amélioration régulière de leur façon de travailler
- Apporter un recul extérieur par rapport aux habitudes installées
- Veiller à réduire le besoin de coordination entre équipes, tout en préservant une culture de collaboration et de partage entre tous



Informations pratiques

Inscription

Les inscriptions peuvent se faire de différentes façons.

Vous pouvez réserver votre place pour la session désirée directement sur Arolla Training (www.arolla.fr/training/), en cliquant sur le bouton « réservez » présent sur chaque page de formation.

Par mail, à ces adresses : contact@arolla.fr

Vous pouvez également nous contacter par téléphone à ce numéro : 01 77 38 82 00

Pour toute demande de réservation, nous vous enverrons un mail de confirmation.

Si vous avez des questions, n'hésitez à nous contacter à ces mêmes coordonnées, nous y répondrons rapidement.

Dédit ou Abandon

En cas de dédit par la société cliente ou d'abandon en cours de formation par un ou plusieurs stagiaires, la société AROLLA remboursera, sur le coût total, les sommes qu'elle n'aura pas réellement dépensées ou engagées pour la réalisation de ladite action.

Prix

Tous les prix indiqués sur notre site sont Hors Taxe.

Organisation de la journée

Les formations se déroulent de 10h à 18h, dans nos locaux du 1er arrondissement de Paris.
Nous vous accueillons dès 9h30 avec un petit-déjeuner offert.
Nous prenons en charge le déjeuner.

Propriété intellectuelle

Les éventuels cours transmis en format PDF ou sur papier ne sont pas autorisés à être communiqués, sauf accord exceptionnel avec le formateur.

Inscription :

↘ en ligne sur www.arolla.fr/training

↘ par e-mail : contact@arolla.fr

↘ par téléphone : 01 77 38 82 00

↘ par courrier : Arolla, 25 rue du Louvre - 75001 PARIS

© Arolla - 2018

Ce catalogue est édité par AROLLA
Crédits photographiques : © Arolla, © Burst
Impression : STIPA